*Journal of The Korea Institute of Information Security & Cryptology* ISSN 1598-3986(Print)
VOL.26, NO.2, Apr. 2016 ISSN 2288-2715(Online)
http://dx.doi.org/10.13089/JKIISC.2016.26.2.301

# 매치메이커: 선호도를 고려한 퍼지 볼트 기법*

툽 신 후,[1†] 강 전 일,[1] 양 대 헌,[1] 이 경 희[2‡]
[1]인하대학교, [2]수원대학교

# Matchmaker: Fuzzy Vault Scheme for Weighted Preference*

Tuvshinkhuu Purevsuren,[1†] Jeonil Kang,[1] DaeHun Nyang,[1] KyungHee Lee[2‡]
[1]INHA University, [2]The university of Suwon

## 요 약

Juels와 Sudan의 퍼지 볼트 기법은 기법이 갖는 오류 내성 때문에 많은 연구에 사용 되어오고 있다. 그러나 이들의 퍼지 볼트 기법은 그들의 논문에서 영화 애호가 문제를 예를 들었음에도 불구하고, 사람들이 일반적으로 갖는 선호도(preference)의 차이에 대한 고려가 존재하지 않는다. 한편, Nyang과 Lee는 안전하고 성능이 좋은 얼굴 인증 시스템을 만들기 위해서, 얼굴 특징이 서로 다른 가중치를 갖도록 얼굴 특징과 퍼지 볼트(vault) 사이에 특별한 연관 구조를 갖는 얼굴 인증 시스템(이른바, 퍼지 얼굴 볼트)을 소개하였다. 그러나 그들의 기법은 일반적인 특징 추출 기법들이 클래스 내부/간 차이를 최적화하려는 특성이 있기 때문에 인증 실패율을 성공적으로 낮추지 못할 것으로 쉽게 예상할 수 있다. 이 논문에서는 퍼지 볼트의 유연성을 제공해주기 위하여 Nyang과 Lee의 퍼지 볼트 기반의 얼굴 인증 시스템에서 가중치 아이디어를 다른 방식으로 구현한 버킷(bucket) 구조와 사용자 선호도와 시스템 구현 간 관계를 공식화하는 세 가지 분포 함수에 대해서 소개한다. 또한 이를 바탕으로 선호도 매치메이커(preference matchmaker) 기법을 제안하며, 영화 데이터베이스를 이용하여 이러한 매치메이커의 연산 성능을 확인해본다.

## ABSTRACT

Juels and Sudan's fuzzy vault scheme has been applied to various researches due to its error-tolerance property. However, the fuzzy vault scheme does not consider the difference between people's preferences, even though the authors instantiated movie lover' case in their paper. On the other hand, to make secure and high performance face authentication system, Nyang and Lee introduced a face authentication system, so-called fuzzy face vault, that has a specially designed association structure between face features and ordinary fuzzy vault in order to let each face feature have different weight. However, because of optimizing intra/inter class difference of underlying feature extraction methods, we can easily expect that the face authentication system does not successfully decrease the face authentication failure. In this paper, for ensuring the flexible use of the fuzzy vault scheme, we introduce the bucket structure, which differently implements the weighting idea of Nyang and Lee's face authentication system, and three distribution functions, which formalize the relation between user's weight of preferences and system implementation. In addition, we suggest a matchmaker scheme based on them and confirm its computational performance through the movie database.

**Keywords:** Reed-Solomon decoding, Berlekamp-Welch algorithm, weighted fuzzy vault, matchmaker

Received(07. 03. 2015), Modified(1st: 11. 27. 2015, 2nd: 03. 31. 2016), Accepted(04. 11. 2016)
* 이 논문은 2014년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행된 것임(NRF-2014R1A1A2059852)
† 주저자, tuvi@isrl.kr
‡ 교신저자, khlee@suwon.ac.kr(Corresponding author)

## I. Introduction

The fuzzy vault, which is introduced by Juels and Sudan in 2002[1], is a useful cryptographic tool. It provides a user to lock her or his secret using a set and another user to unlock the secret using another set if two sets are sufficiently overlapped. In their paper, the authors instance the movie lover who wants to find someone with similar preference. Due to its error-tolerance property, the fuzzy vault scheme has been utilized in various researches, especially in biometrics area [2-11].

In 2007, Nyang and Lee proposed a face authentication system, so-called fuzzy face vault, based on the fuzzy vault scheme[12]. In the system, the authors introduce the concept of weighted features. Features, for a face, are represented as a vector and used for comparing with other face features. In the comparison, some methods of extracting features from faces such as principal component analysis (PCA) and linear discriminant analysis (LDA) use geometric distances. In a feature vector, there are significant or less significant features. Therefore, when the fuzzy vault scheme is applied to face authentication system, the authentication failure (in terms of false acceptance or false rejection) may increase if one feature is mapped to only one point. To compensate the loss of significance, the method of weighting features is essential in [12]. However, their face authentication scheme does not seem to decrease the authentication failure because it still uses geometric distances to find correct features and chaff features are located within narrow ranges.

In this paper, we newly suggest a weighted fuzzy vault scheme. Our contributions in this paper are as follows: 1) We introduce 'bucket' structure for implementing the weighting idea in a different manner for the ordinary fuzzy vault scheme. By doing so, we can make the fuzzy vault scheme to be used not only in equal preference environments, but also in weighted preference environments. 2) We propose three distribution functions for guaranteeing the flexible use of the fuzzy vault scheme to various applications. They formalize the relation between user's preference and system implementation, so that they directly affect the usability and security of the system. As an example of our proposal, we implement the movie matchmaker system.

The rest of this paper is organized as follows. In Section 2, we briefly address the polynomial and Reed-Solomon (RS) error correction code. In Section 3, we analyze the fuzzy face vault. In Section 4, we introduce our proposal, matchmaker. The computational performance of the matchmaker is shown in Section 5. In Section 6, we discuss some issues related the matchmaker. Section 7 includes the conclusion.

## II. Preliminaries

### 2.1 Polynomial over Galois field

Galois field $\mathbb{F}_{p^r}$ is field that has finite elements with the order $p^r$, where $p$ is a prime number and $r$ is a natural number. Each element in $\mathbb{F}_{p^r}$ can be represented to a vector such as $(a_1,a_2,\cdots,a_r) \in (\mathbb{Z}_p)^r$. When $r=1$ and $p \neq 2$, $\mathbb{F}_p$ is often referred to as 'prime field.' When $r \neq 1$ and $p=2$, $\mathbb{F}_p$ is often referred to as 'binary field.'

Over a Galois field $\mathbb{F}$, a polynomial

$P(x) = m_0 + m_1 x + \cdots + m_{k-1} x^{k-1}$ can be defined as a set of points $\{(x_i, P(x_i))\}_{x_i \in \mathbb{F}}$. For example, $P(x) = 4x^2 + 3x + 2$ over $\mathbb{F}_7$ can be determined as $\{(0,2),(1,2),(2,3),(3,5),(4,1),(5,5),(6,3)\}$. If we gather $k$ or more points on $P(x)$, we can reconstruct $P(x)$ though the Gaussian elimination.

### 2.2 Reed-Solomon error-correcting code

RS code is a group of error-correcting codes[13]. It is able to detect and correct multiple errors from the received code word. As shown in Fig.1, there is RS code with parameters $q$, $n$, and $k$: the number of all possible elements $q$, a code word length $n(\leq q)$, and a message length $k(\leq n)$. Each element is interpreted as Galois field $\mathbb{F}$.

As the original view of RS code, our intention is to interpret an original message $m = (m_0, m_1, \cdots, m_{k-1})$ as coefficient of a certain polynomial $P(x) = \sum_{i=0}^{k-1} m_i x^i$ over $\mathbb{F}$. To compute the code word, $P(x)$ is evaluated at $n$ distinct elements $(x_1, x_2, \cdots, x_n)$. The code word is equal to $\{P(x_1), P(x_2), \cdots, P(x_n)\}$. If $(x_1, x_2, \cdots, x_n)$ are unknown, the code word should be represented as the set of points such as $\{(x_1, P(x_1)), (x_2, P(x_2)), \cdots, (x_n, P(x_n))\}$.

To decode the code word, many algorithms were introduced: Berlekamp-Welch[14], Berlekamp-Massey[15], Euclid's algorithm[16], Gao[17] and so on. In this paper, we used Berlekamp-Welch algorithm for our experiments.

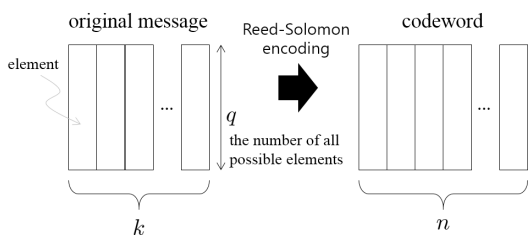



Fig. 1. Reed-Solomon encoding and its parameters

In $[k,n,q]$ Berlekamp-Welch algorithm, the upper bound of errors that can be corrected is less than $(n-k+1)/2$. In other word,

$$0 < e < \frac{n-k+1}{2}, \quad (1)$$

where $e$ denotes the number of errors. Berlekamp-Welch algorithm returns a non-zero polynomial $P(x)$ of degree at most $k-1$.

To recover $P(x)$, Berlekamp-Welch algorithm first computes non-zero error locator polynomial $E(x)$ of degree $e$ and $Q(x)$ of degree $(e+k-1)$, and computes $P(x) = Q(x)/E(x)$. Computing $E(x)$ and $Q(x)$ are as difficult as computing $P(x)$. While each of these polynomials are difficult to find individually, the pair of polynomials $(E(x), Q(x))$ can be found in polynomial time (i.e., $O(n^3)$). Berlekamp-Welch algorithm successfully returns $P(x)$ if $E(x)$ divides $Q(x)$ without any remainder.

### 2.3 Symbols and their explanations

In this paper, we use the following symbols show in Table 1. for simplicity of description.

## III. Nyang and Lee's Fuzzy Face Vault

In the fuzzy vault scheme, every preferences have equal strength. In other words, each preference is transformed a certain amount of points on the secret polynomial. Therefore, the fuzzy vault scheme cannot be directly applied to the

Table 1. Symbols and their explanations

| | explanation |
|---|---|
| Reed-Solomon code parameters | |
| $\mathbb{F}$ | Galois field |
| $k$ | the size of message |
| $n$ | the size of code word |
| $q$ | the order of Galois field |
| $e$ | the number of maximum errors |
| $P(x)$ | (secret) polynomial |
| distribution function | |
| $\Omega(i)$ | weight distribution |
| $\Gamma(i)$ | chaff distribution |
| $\Theta(i)$ | code word distribution |
| set | |
| $B_i$ | $i$-th bucket, a set of preferences |
| $C_i$ | a set of counterfeits in $B_i$ |
| $Z_i$ | a set of points derived from $B_i$, $Z_i^{fav}$ and $Z_i^{cnt}$: a set of points derived from favorite and counterfeits in $B_i$, $Z_i = Z_i^{fav} \cup Z_i^{cnt}$ |
| $X_i$ | a set of x-coordinates derived from $B_i$, $X_i^{fav}$ and $X_i^{cnt}$: a set of x-coordinates derived from favorite and counterfeits in $B_i$, $X_i = X_i^{fav} \cup X_i^{cnt}$ |
| etc. | |
| $v$ | the number of user's favorites |
| $v'$ | the number of buckets |
| $u$ | user information such as name, telephone number, and address |
| $h()$ | (cryptographic) hash function |
| $R(x)$ | random element generator, $R(x) \neq P(x)$ |
| $Enc()$ | symmetric encryption function |
| $Dec()$ | symmetric decryption function |

environments that different strength of preferences should be considered.

In 2007, Nyang and Lee introduced a face authentication system based on the fuzzy vault, so-called the fuzzy face vault[12]. In their paper, the authors illustrated their scheme in the face verification system. Different from the fuzzy vault scheme, the fuzzy face vault has two-layered structure: it consists of intermediate and coordinate layer.

In the intermediate layer, a single captured feature (e.g., an element of a feature vector) is transferred to several number of X-Y coordinates. In the coordinate layer is created by RS code word representing a secret as a polynomial $P(x)$ as the ordinary fuzzy vault does.

### 3.1 Locking and unlocking procedures

In the paper of Nyang and Lee[1], the features are obtained by using a classifier (e.g., PCA or LDA) from facial images. The weights of the features can be proportionally decided according to the distribution of features' differences.

Let $F = \{f_1, f_2, \cdots\}$ be a set of genuine features. To lock a vault in the fuzzy face vault scheme, a feature $f_i$ with a certain weight $w_i$ is reconstructed as

$$X_i = \{h(f_i \| x) \in \mathbb{F} | 1 \leq x \leq w_i\} \quad (2)$$

where $h()$ denotes one way and collision free hash function. And then, the system randomly generates a secret polynomial $P(x)$. The system stores a set of points $\{(x, P(x)\}_{x \in X_1 \cup X_2 \cup \ldots}$ with chaff features on the intermediate layer and chaff points on the coordinate layer. Note that every chaff points should be matched to certain chaff features. As the result, higher weighted features are mapped into more points.

To unlock the vault, the user inputs her or his features. As doing similar task with the locking procedure, the system collects the points on the coordinate layer. By using RS decoding algorithm, $P(x)$ can be recovered from the collected points when the number of errors caused mistakenly capturing is less than a certain threshold.

### 3.2 Difficulty of implementation

To determine feature on the intermediate layer, the fuzzy face value uses the geometric distance measurement such as Euclidean and Manhattan distances. Therefore, too many chaffs on intermediate layer are not desirable because the distance between genuine and chaff feature may be closer than threshold for error tolerance.

In PCA, for example, the distribution of features' differences (between maximum and minimum values) seems to be lied on exponential curve. It means that matching on the most significant feature (on the intermediate layer) may derive the half of genuine points (on the coordinate layer) to reconstruct the secret polynomial. Thus, to guarantee the minimum level of security (e.g., 1/10,000), the system should add more chaffs for more significant features. In this case, the system may not correctly find significant features even user correctly input her or his genuine facial image. Even if two values are really similar, their hashed values are totally differentiated. Therefore, chaff features must not be located within reasonable error bound (in terms of differences of inter-class and/or intra-class). Considering feature extraction methods optimize the differences, the fuzzy face vault does not seem to work with the facial verification and authentication system because we cannot avoid the chaff features to be located within error bound (i.e., difference of intra-class).

## IV. Our Proposal: Weighted Fuzzy Vault

Even though Nyang and Lee's fuzzy face vault scheme does not seem to work as their expectation, the weighting idea is reasonable. In this paper, rather than improving the fuzzy face vault, we generalize the fuzzy vault to cover various applications by implementing weighting idea in a different manner. As one of applications, we introduce the matchmaker, which helps people to find out other people who have the similar preferences without revealing their preferences.

### 4.1 Overview

People may have different preference in different issues or areas. Someone who has a big concern about movies may not have any concern about sports stars. Even though two girls like the same celebrities, their most favorite celebrities may be different. A question may arise when we use the ordinary fuzzy vault for checking their preferences: can we say that they have the similar preference? To answer the question, we can make the following system, so called 'matchmaker.'

The matchmaker consists of two procedures: template making and user searching. In the template making procedure, a user must offer their favorites with certain weight values. For example, Alice may input "Alice in wonderland" with weight value 10 and "Harry Potter and the Half-Blood Prince" with weight value 2. The matchmaker system makes a Alice's template and stores it. In the user searching procedure, another user also must offer their favorites without weight values. For example, Bob may input "Alice in wonderland" and "Harry Potter and the Potter and Chamber of secret." The matchmaker system compares Bob's favorite movies with all templates stored in the system. In this example, the system

is likely to find Alice.

To generalize the face vault, we newly introduce the concept of preference buckets $B=\{B_1,B_2,\cdots\}$, weight distribution $\Omega(i)$, chaff distribution $\Gamma(i)$, and code word distribution $\Theta(i)$. Each preference bucket is filled a genuine favorite and a huge number of counterfeits and the number of buckets depends on the number of user's favorites. The weight distribution is defined by the weight values from users, but we assume that the weight distribution follows a certain well-known distribution such as linear, exponentiation, and normal distribution (i.e., S-curve). The chaff distribution indicates the number of counterfeits in each bucket for the security reason. The code word distribution means how many points should be generated from a single favorite or counterfeit in a bucket. The chaff and code word distributions depend on the weight distribution.

In the following section, we explain in detail how the matchmaker works with movie scenario as illustrated in the ordinary fuzzy vault scheme.

### 4.2 How to make preference template

Let $n$ be the size of cord word, $k$ be the size of message, and $q$ be the order of Galois field $\mathbb{F}$ as parameters of RS error correction code. Basically, $k \le n \le q$. Then, $n$ should be equal to $\sum_{i=1}^{v}\Omega(i)$, $(k-1)$ should be the degree of the secret polynomial $P(x)$.

Alice suggests her favorite movies set $\{(m_i,w_i)\}_{i\in[1..v]}$ to the matchmaker system, where $m_i$ is movie name, $w_i$ is weight of movie, and $v$ is the number of favorite movies. Note that the movies' weight follows the weight distribution $g(i)$ (i.e., $w_i=\Omega(i)$).

The system randomly generates a secret $s=(s_0,s_1,\cdots,s_{k-1})$, computes $S=h(s)$, and interprets as secret polynomial $P(x)=\sum_{i=0}^{k-1}s_ix^i$. Alice's personal information $u$ is encrypted by using $s$ such that $U=Enc(s,u)$. Each favorite movie $m_i$ is classified into each preference bucket $B_i$ (i.e., $m_i\in B_i$). When two or more movies have the same weight, they should be classified into the same bucket. Thus the number of buckets $v'$ is less than or equal to $v$.

Let $C_i$ be a set of counterfeits for bucket $B_i$. According to the chaff distribution $\Gamma(i)$, the system adds counterfeits into each bucket. Then, $B_i=C_i\cup\{m_i\}$, $|C_i|=\Gamma(w_i)$, and $C_i\cap C_j=\varnothing$ if $i\neq j$. After that, the system shuffles all bucket for hiding the favorite movies and computes x-coordinates for all movies (including favorite and counterfeit movies) in each bucket such that

$$X_i^{fav}=\{h(m_i\|x)\,|\,1\le x\le\Theta(w_i)\}\ \text{and} \quad (3)$$

$$X_i^{cnt}=\{h(c_j\|x)\,|\,1\le x\le\Theta(w_i),c_j\in C_i)\}. \quad (4)$$

For all x-coordinates in $X_i^{fav}$ and $X_i^{cnt}$, the system evaluates the secret polynomial such that

$$Z_i^{fav}=\{(x,P(x))|x\in X_i^{fav}\}\ \text{and} \quad (5)$$

$$Z_i^{cnt}=\{(x,y)|x\in X_i^{cnt}\wedge x\notin X_i^{fav}\wedge y=R(x)\}, \quad (6)$$

where $R()$ denotes a random element generator avoiding $P(x)$. Note that $|Z_i=Z_i^{fav}\cup Z_i^{cnt}|\le|B_i|\times\Theta(w_i)$ because of the hash collision and all points in

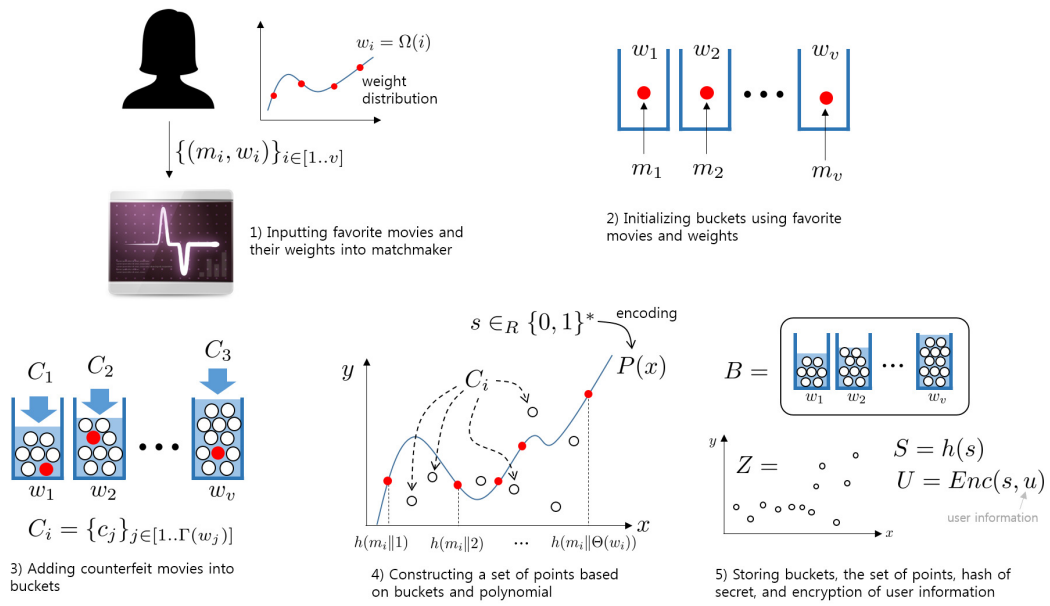



Fig. 2. Procedure for making a movie preference template

$Z = \bigcup_{i=1}^{v} Z_i$ have different x-coordinates and the number of points in $Z$ cannot exceed $q$.

Finally, the system store

$$\langle \{(B_i, w_i)\}_{i=[1..v']}, Z, S, U \rangle \quad (7)$$

as Alice's preference template. This procedure is illustrated in Fig.2.

Note that we cannot directly apply Nguyen *et al*'s technique[11] to generate chaff points because they are generated from the counterfeit movies. The proposed system makes the finding collision (i.e., polynomial) difficult by using the cryptographic hash function for checksum instead of using cyclic redundancy check (CRC) as many fuzzy vault-based biometrics systems do.

### 4.3 How to find people

To find people who have similar preference, Bob inputs his favorite movies $M = \{m_i'\}_{i\in[1..v]}$ to the matchmaker system. Given a preference template $\langle \{(B_i, w_i)\}_{i=[1..v']}, Z, S, U \rangle$, the system searches each movie $m_i'$ in all buckets $\{B_1, \cdots, B_{v'}\}$ and finds out the corresponding weight $w_i'$. If the system cannot find $m_i'$ in any bucket, it removes $m_i'$ from $M$ (i.e., $M = M - \{m_i'\}$). For each $m_i'$ with $w_i'$, the matchmaker computes x-coordinates such that

$$X_i' = \begin{cases} \{h(m_i'\|x) | 1 \le x \le \Theta(w_i')\} & \text{if } m_i' \in M \\ \varnothing & \text{if } m_i' \notin M \end{cases} \quad (8)$$

And then, for each x-coordinate in $X' = X_1' \cup \cdots \cup X_{v'}'$, the system collects a corresponding point in $Z$ (i.e., the points in $Z$ whose x-coordinates are identical to the x-coordinates in $X'$).

If the number of collected points is greater than $k-1$ and less than $(n+e+1)$, the system tries to reconstruct secret

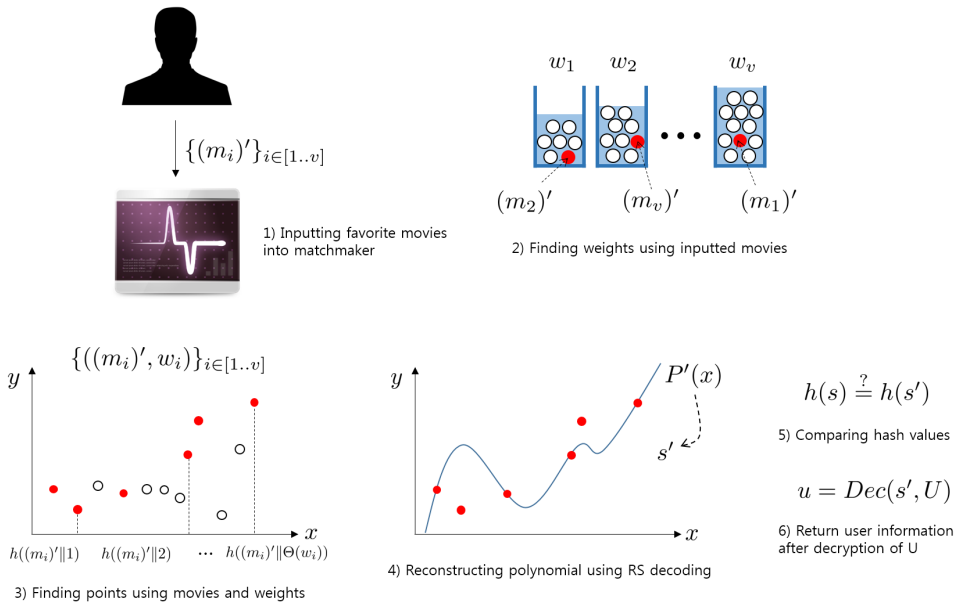



Fig. 3. Procedure for searching people who have similar movie preference

polynomial $P(x)$ by using Berlekamp-Welch algorithm. If the Berlekamp-Welch algorithm returns $P'(x)$, it extracts the coefficients $s' = (s_0', s_1', \cdots, s_{k-1}')$ and computes $S' = h(s')$. If $S = S'$, the system notifies user's information of current preference template to Bob after decrypting $U$ such that $Dec(s', U)$. And then, the system continues the searching procedure to the next user's preference template. This procedure is shown in Fig.3.

### 4.4 Security Parameters

In the fuzzy vault scheme[1], an attacker who wants to reveal the locked secret (as a corresponding polynomial) is mainly concerned. To guarantee the sufficient security level against that attacker, the matchmaker system should carefully choose the parameters and distributions.

If the attacker can choose $k$ or more genuine points from $Z$, it can reconstruct the secret polynomial. This probability $p_1$ is equal to $C(|Z|,k)^{-1}$ and the total number of points in $Z$ is slightly less or equal to $\sum_{i=1}^{v'}|B_i|\times\Theta(w_i)$. On the other hand, the number of elements in bucket $B_i$ is equal to $\Gamma(w_i)+1$. Therefore,

$$p_1 = C(|Z|,k)^{-1} \approx C(\min(q, \sum_{i=1}^{v'}\Gamma(\Omega(i))\Theta(\Omega(i))),k)^{-1} \quad (9)$$

Since each element is linked to the points in $O$, the attacker may reconstruct the secret polynomial by choosing $v$ or less elements (e.g., movies) in the buckets. If $\Gamma(i)$ is linear distribution, the attacker must choose elements (e.g., movies) from higher weighted buckets (one element in one bucket) so that the number of linked points in $Z$ is greater than $(k-1)$ and less than $(n+e+1)$. Let $\tau$ be the minimum number of elements that the attacker

should choose. Then, $\tau \le v'$ and $(n+e+1) > (w_{v'} + w_{v'-1} + \cdots + w_{v'-\tau-1}) \ge k$. The probability of this attack ($=p_2$) is equal to $1/\prod_{i=v'-\tau-1}^{v'} |B_i|$. Therefore,

$$p_2 \approx \left( \prod_{i=v'-\tau-1}^{v'} \Gamma(\Omega(i)) \right)^{-1}. \quad (10)$$

Obviously, $p_1 < p_2$ in most cases. Due to the variety of definitions of distributions, in this paper, we offer a few parameter instances with its security level. Note that $q$ is not deeply related to security strength except hash collision problem.

**Example 1)** If $v = v' = 10$, $k = 31$, $q = 104729$, $\Omega(i) = i$, $\Gamma(i) = a_1 i + b_1$, and $\Theta(i) = a_2 i + b_2$, then $n = 55$ and $e \le 12$. In addition, if we set $a_2 = 1$ and $b_2 = 0$, then $\tau = 4$ ($\because \Theta(10) + \Theta(9) + \Theta(8) + \Theta(7) \ge k = 31$). In this case, the probabilities $p_1$ and $p_2$ are approximately close to

$$p_1 \approx 1/C(\sum_{i=1}^{10} \Gamma(i), 31) \text{ and} \quad (11)$$

$$p_2 \approx 1/(\Gamma(10)\Gamma(9)\Gamma(8)\Gamma(7)). \quad (12)$$

If $a_1 = 100$ and $b_1 = 100$, $p_1 \approx 2^{-280}$ and $p_2 \approx 2^{-40}$.

**Example 2)** If $v = v' = 5$, $k = 31$, $q = 104729$, $\Omega(i) = i$, $\Gamma(i) = 1000i + 1000$, and $\Theta(i) = 3i + 3$, then $n = 60$, $e \le 14$, and $\tau = 2$ ($\because \Theta(5) + \Theta(4) \ge k = 31$). In this case, $p_1 \approx 1/C(q, 31) \approx 2^{-404}$ and $p_2 \approx 1/(\Gamma(5)\Gamma(4)) \approx 2^{-28}$.

As shown in the above examples, when $v$ is relatively small, it is difficult to achieve higher level of security even with the huge number of counterfeits.

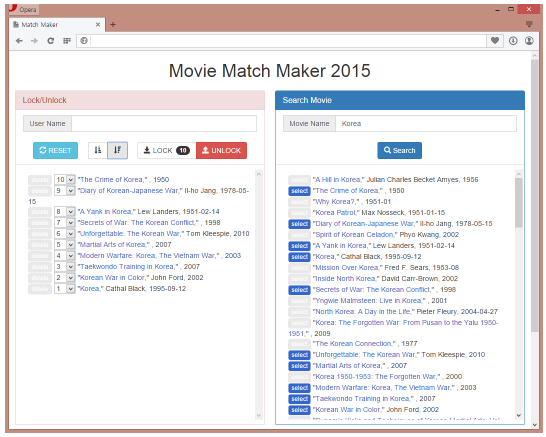



Fig. 4. User interface of movie matchmaker

## V. Experiments

### 5.1 Experiment environment

To confirm the overall performance of our proposal, we implemented the movie matchmaker as illustrated in Section 4. For experiments, we implemented a server program using Python 2.7.3 on Ubuntu 12.04.4 x64 Server running on Intel Xeon E5-2620@2.00GHz CPU with 64GB RAM and a user interface program using HTML5 (with JavaScript) as shown in Fig.4. We collected 266,263 movies (i.e., title, director, release date, etc.) from Freebase database powered by Google and stored them using MongoDB 2.4.14. We applied two type of hash functions: Python built-in hash function for mapping movies to x-coordinates and SHA-1 for computing the hash value of secret polynomial.

We performed experiments of two parameter examples as described in Section 4.4. In each parameter, we measured times for making a template and searching people. Specifically, in searching people, we stored only one template in database and measured the various cases

Table 2. Number of elements in $B$ and $Z$

| | Example 1 | Example 2 |
|---|---|---|
| $\|B\|$ | 6,510 | 65,005 |
| $\|Z\|$ | $\leq$ 44,055 | $\leq$ 104,729($=q$) |

that made different code word size*. Table 2 shows the number of elements in $B$ and $Z$. Each experiment was repeated in 100 times.

### 5.2 Experiment results

Fig.5 shows the response time for making a template. 4.841$s$ and 9.735$s$ respectively took in example 1 and 2 on median. The number of hash operations to map movies to x-coordinates is equal to 44,055 in example 1 and 870,060 in example 2. On the other hand, the number of polynomial evaluations is exactly same with the size of $Z$: 44,055 in example 1 and 104,729 in example 2. The gaps between example 1 and 2 are about 20 times in hash operations and about 2 times in polynomial evaluations. Therefore, we can conclude that most significant time consuming occurs when the system evaluates the secret polynomial for computing points in $Z$.

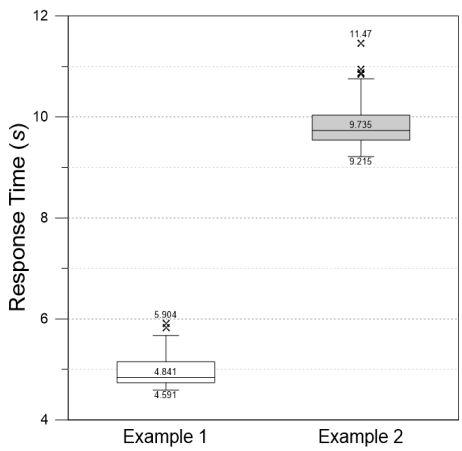
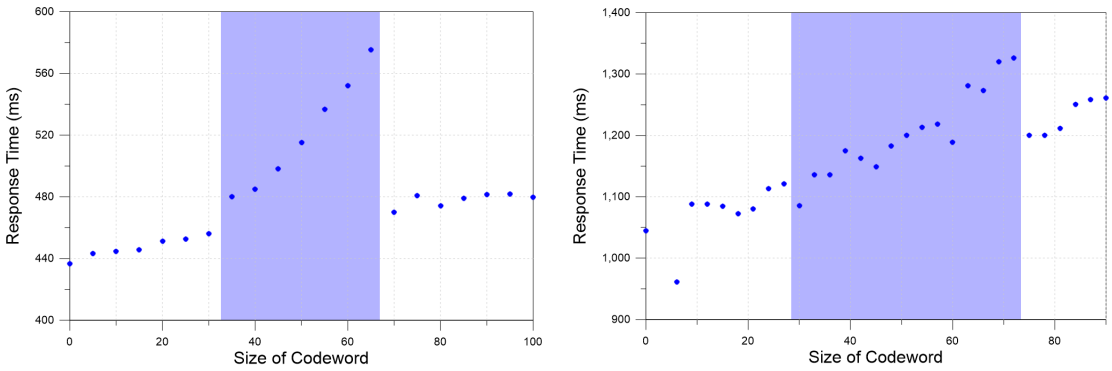



Fig. 5. Response times for making a template

Fig.6 shows the average response times for finding people who have similar movie preference. In our experiments, the response times are lied between 360$ms$ and 625$ms$ in example 1 and between 904$ms$ and 1,742$ms$ in example 2. As the size of code word (generated according to user's inputs) increases, the overall time also increases. When the size of code word meets the condition, which is described in Section 4.3, the system runs Berlekamp-Welch algorithm. Note that there is no big difference of response times between when Berlekamp-Welch algorithm returns fail and secret polynomial's coefficients. When Berlekamp-Welch algorithm runs, the response times slightly increase (about 50~100$ms$) even though its time complexity is $O(n^3)$. Moreover, the number of hash operations for mapping movies to x-coordinates is the same to the size of code word; the time consumption for hashing is not that much. Therefore, the most significant time consuming occurs due to searching movies in buckets.

In the experiments, the server program utilizes 'in' operation of Python to search movies in buckets. This operation is known to have $O(n)$ time complexity. However, if we use the tree mechanism, we can reduce the searching time to $O(\lg n)$ time. In addition, the matchmaker system includes a lots of parts that the parallel processing can be applied to. For example, the hash operations for mapping movies to x-coordinates and the polynomial evaluation can be independently proceeded.

* According to the weights of inputted movies, the size of code word varies. For example, a user may input several movies classified in one bucket.




(a) Example 1 (b) Example 2

Fig. 6. Average response times for finding people who have similar movie preference. Blue areas mean the matchmaker additionally runs Berlekamp-Welch algorithm.

## VI. Other issues

### 6.1 Polynomial reconstruction by adversary

RS error correction provides the way to reconstruct the secret polynomial even with some errors. The capability of error correction is proportioned to the gap between the size of code word and original message such that $e < (n-k+1)/2$. However, $e$ is almost half of $(n-k)$. In other words, to correct $e$ errors, additional $e$ genuine points are required. Therefore, as described in Section 4.4, the attacker who chooses only $k$ points takes more advantage than who chooses more $k$ points unless the probability of which it chooses genuine points exceeds 0.5.

### 6.2 Preference similarity

In this paper, we simply assume that the preference similarity is close to $k/n$ as the fuzzy vault does. However, defining of similarity is more complicated than our intuition. In many areas such as biometrics, the similarity is checked by using geometric distances, but people's preferences are difficult to be represented as vectors due to various reasons such as ignorance and disliking. People may not even know most movies' names or may dislike (or hate) some movies. Even though the favorite movies of Alice and Bob are exactly same, but the most favorite movies may be different. We think the weighted matching method is much better than simple matching method, but the former still does not even consider the above situation.

We remain this issue as our further works. To do this, we should deeply consider what preference is and develop (or research) suitable methods of comparing preferences. After that, we will try to implement advanced matchmaker system dealing with dynamic user preferences in terms of the number of favorites and their weights.

### 6.3 Personal entropy system

In the fuzzy vault scheme, the personal entropy system is mentioned as one of useful applications. The personal entropy system provides system users to recover their secrets[18]. In the personal entropy system, a secret is divided into several partial secrets (by using the secret sharing scheme) and a trusted third party

stores the partial secrets with personal questions such as "When is your mother's birthday?" If a user can answer sufficient questions, she or he can recover their secret.

The matchmaker can be easily converted to the personal entropy system. Instead of answering personal questions, users are required to input their preferences. As time goes on, users' preferences may change, but highly weighted items perhaps remain in their preferences.

However, to convert the matchmaker to the personal entropy system, $k$, the degree of secret polynomial, and $n$, the size of code word, should be reduced to the reasonable level. In example 1 described in Section 4.4, for instance, the system must attempt up to $4.65\times10^{17}(\approx C(n+e,k))$ cases (i.e., secret recovering in secret sharing scheme) in order to reconstruct the polynomial if RS decoding fails. Instead of reducing $k$ and $n$, much more counterfeits are required. It will cause the increase of time consumption for making templates. Fortunately, the procedure for making template is required only one time for each user, and thus, it is not a big problem to consider.

### 6.4 Setting for ordinary fuzzy vault

As we mentioned in Section 4, the matchmaker generalizes the fuzzy vault. We can implement the ordinary fuzzy vault based on the matchmaker by adjusting distributions as $\Omega(i)=1$, $\Gamma(i)=b_2$, and $\Theta(i)=1$ where $b_2$ denotes the number of chaffs in a bucket. In the template, there is only one bucket and all chaffs and preferences are located in that bucket. If $q$ is large enough, mapping from an element in $B$ to a point in $Z$ is almost bijective (one-to-one correspondent).

## VII. Conclusion

In this paper, we eliminate the geometric distance measurement in the fuzzy face vault scheme and generalize the fuzzy vault scheme for various applications. As one of applications, we introduce the matchmaker. By adopting the bucket concept and three different distributions (i.e., weight, chaff, and code word distributions), we let the matchmaker be able to cover not only movies but also various preferences. Though the experiments, we confirm the overall performance of the matchmaker under two different parameter settings. To use the matchmaker in the real world, various speed-up techniques are essential.

For our future works, we will develop advanced matchmaker with better performance to deal with dynamic user preferences. In addition, we want to implement the personal entropy system based on the advanced matchmaker. By performing user experiments on that system, we will try to confirm the appropriateness of our approach.

## References

[1] A. Juels and M. Sudan, "A fuzzy vault scheme," Proceedings of IEEE International Symposium on Information Theory (ISIT), p. 408, Jun 2002.

[2] K. Nandakumar, A.K. Jain, and S. Pankanti, "Fingerprint-Based Fuzzy Vault: Implementation and Performance," IEEE Transactions on Information Forensics and Security, vol. 2, no. 4, pp. 744-757, Dec. 2007.

[3] A.-Y. Kim and S.-H. Lee, "Authentication Protocol using Fuzzy Eigenface Vault based on MoC," Proceedings of International Conference on Advanced

Communication Technology, vol. 3, pp. 1771-1775, Feb. 2007.
[4] G.X. Qiao and H.A. Qun, "The Automatic Fuzzy Fingerprint Vault Based on Geometric Hashing: Vulnerability Analysis and Security Enhancement," Proceedings of 2009 International Conference on Multimedia Information Networking and Security (MINES), vol. 1, pp. 18-20, Nov. 2009.
[5] D. Moon, S. Lee, Y. Chung, S.B. Pan, and K. Moon, "Implementation of automatic fuzzy fingerprint vault," Proceedings of International Conference on Machine Learning and Cybernetics, vol. 7, pp. 3781-3786, Jul. 2008.
[6] S. Lee, and D. Moon, H. Choi, and Y. Chung, "Memory-Efficient Fuzzy Fingerprint Vault based on the Geometric Hashing," Proceedings of International Conference on Information Security and Assurance (ISA), pp. 312-315, Apr. 2008.
[7] D. Moon, W. Choi, K. Moon, and Y. Chung, "Fuzzy fingerprint vault using multiple polynomials," Proceedings of IEEE International Symposium on Consumer Electronics (ISCE), pp. 290-293, May 2009.
[8] L. Wu, and S. Yuan, "A Face Based Fuzzy Vault Scheme for Secure Online Authentication," Proceedings of International Symposium on Data, Privacy and E-Commerce (ISDPE), pp. 45-49, Sep. 2010.
[9] V. Joshi and P. Sanghavi, "Three tier data storage security in cloud using Face fuzzy vault," Proceedings of International Conference on Computing, Communication and Applications (ICCCA), pp. 1-6, Feb. 2012
[10] D. Moon, Y. Chung, C. Seo, and S.Y. Kim, "A practical implementation of fuzzy fingerprint vault for smart cards," Journal of intelligent Manufacturing, vol. 25, pp. 293-302, Apr. 2014.
[11] M.T. Nguyen, Q.H. Truong, and T.K. Dang, "Enhance fuzzy vault security using nonrandom chaff point generator," Information Processing Letters, vol. 116, no. 1, pp. 53-64, Jan. 2016.
[12] D. Nyang and K. Lee, "Fuzzy Face Vault: How to Implement Fuzzy Vault with Weighted Features," Proc. of International Conference on Human-Computer Interaction, HCII 2007, LNCS 4554, pp. 491-496, 2007.
[13] I.S. Reed and G. Solomon, "Polynomial Codes over Certain Finite Fields," Journal of the Society for Industrial and Applied Mathematics (SIAM), vol. 8, no. 2, pp. 300 - 304, 1960
[14] J.L. Massey, "Shift-register synthesis and BCH decoding," IEEE Transactions on Information Theory, vol. IT-15, no. 1, pp. 122 - 127, 1969.
[15] L.R. Welch and E.R. Berlekamp, "Error Correction for Algebraic Block Codes," US 4,633,470, Dec. 30, 1986.
[16] Y. Sugiyama, M. Kasahara, S. Hirasawa, and T. Namekawa, "A method for solving key equation for decoding Goppa codes," Information and Control, Vol.27, pp. 87 - 99, 1975.
[17] S. Gao, "A new algorithm for decoding Reed-Solomon codes," Proceedings of Communications, Information and Network Security, pp. 55-68, Dec. 2002.
[18] C. Ellison, "Emergency Key Recovery without Third Parties," talk given at the Crypto '96 rump session, Aug. 1996.

## 〈저자소개〉

툽 신 후 (Tuvshinkhuu Purevsuren) 학생회원
2012년 6월: 푸네대학교 정보공학과 학사
2012년 7월~2013년 6월: Aravt Technology
2016년 2월: 인하대학교 컴퓨터정보공학과 석사
2016년 3월~현재: 브이에스코리아
〈관심분야〉 개인정보보호, 컴퓨터비전

강 전 일 (Jeonil Kang) 정회원
2003년 2월: 인하대학교 전기전자컴퓨터공학과 학사
2006년 2월: 인하대학교 정보통신대학원 석사
2014년 8월: 인하대학교 정보통신공학과 박사
2014년 9월~현재: 인하대학교 인간중심컴퓨팅연구소 박사후연구원
〈관심분야〉 RFID 보안, 생체 인식 보안, 무선 센서 네트워크 보안, 무선 인터넷 보안, 웹 인증 보안

양 대 헌 (DaeHun Nyang) 종신회원
1994년 2월: 한국과학기술원 과학기술대학 전기 및 전자공학과 학사
1996년 2월: 연세대학교 컴퓨터과학과 석사
2000년 8월: 연세대학교 컴퓨터과학과 박사
2000년 9월~2003년 2월: 한국전자통신연구원 정보보호연구본부 선임연구원
2003년 2월~현재: 인하대학교 컴퓨터정보공학과 교수
〈관심분야〉 암호이론, 암호 프로토콜, 인증 프로토콜, 무선 인터넷 보안

이 경 희 (KyungHee Lee) 정회원
1993년 2월: 연세대학교 컴퓨터과학과 학사
1998년 8월: 연세대학교 컴퓨터과학과 석사
2004년 2월: 연세대학교 컴퓨터과학과 박사
1993년 1월~1996년 5월: LG소프트(주) 연구원
2000년 12월~2005년 2월: 한국전자통신연구원 선임연구원
2005년 3월~현재: 수원대학교 전기공학과 부교수
〈관심분야〉 바이오인식, 정보보호, 컴퓨터비전, 인공지능, 패턴인식